Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	302	717/106.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/05/23 08:35
L2	230	717/118.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/05/23 08:35
L3	287	717/130.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/05/23 08:35
L4	427	717/140.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/05/23 08:36
L5	215	717/148.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/05/23 08:36
L6	250	717/158.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/05/23 08:36
L7	41	((byte adj code) or byte-code or bytecode) same (native or jit or jit$4) and (generat$3 or creat$3 or produc$3 or provid$3) near3 (instrument$5 or prolog$2 or epilog$2)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:37
L8	132	((byte adj code) or byte-code or bytecode or java) same (native or jit or jit$4) and (generat$3 or creat$3 or produc$3 or provid$3) near3 (instrument$5 or prolog$2 or epilog$2)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:39
L9	1	l1 and l8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:38

L10	6	l2 and l8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:37
L11	12	l3 and l8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:37
L12	2	l4 and l8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:38
L13	9	l5 and l8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:38
L14	6	l6 and l8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:38
L15	105	((byte adj code) or byte-code or bytecode or java) same (native or jit or jit$4) and (generat$3 or creat$3 or produc$3 or provid$3) near3 (instrument$5 or prolog$2 or epilog$2) and (guard$3 or inhibit$3 or prevent$3 or disabl$3 or boolean or conditional)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:40
L16	1	l1 and l15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:40
L17	2	l2 and l15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:41
L18	9	l3 and l15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:42

L19	2	l4 and l15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:46
L20	3	l5 and l15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:46
L21	2	l6 and l15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/05/23 08:40
S1	1150	debug$4 and (re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/02/16 12:36
S2	196	debug$4 and (re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again))) and (watch or watchpoint or value or variable) and native and (instrument$5 or hook$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/30 15:00
S3	17	debug$4 and (re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again))) and (watch or watchpoint or value or variable) and native same (instrument$5 or hook$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/05 18:18
S4	53	debug$4 and (re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again))) and (watch or watchpoint) and native and (instrument$5 or hook$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/02/05 18:31
S5	3	debug$4 and (re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again))) and (watch or watchpoint) and native same (instrument$5 or hook$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/02/05 18:30
S6	11	debug$4 and (watch or watchpoint) and native same (instrument$5 or hook$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/02/05 18:31
S7	188	717/130.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/02/16 12:36

S8	301	717/124.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/02/16 12:36
S9	305	dynamic$4 same ((compil$5 or recompil$5 or re-compil$5) and native)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/02/16 12:39
S10	19	dynamic$4 same ((compil$5 or recompil$5 or re-compil$5) and native) and (watch or watchpoint or watch-point or (data near2 breakpoint))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 13:07
S11	22	jvmdi or ((debug near3 interface) same (java or bytecode or byte-code or (byte adj code)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 13:09
S12	31	jvmdi or ((debug$4 near3 interfac$3) same (java or bytecode or byte-code or (byte adj code)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 13:31
S13	4	generat$3 near3 (instrument$5 or hook) same native and 717/???.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 13:35
S14	13	generat$3 near3 (instrument$5 or hook) same native	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 13:39
S15	13	generat$3 near3 (stub) same native	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 13:39
S16	1	(generat$3 near3 (stub) same native) and (generat$3 near3 (instrument$5 or hook) same native)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 13:39
S17	12	(generat$3 near3 (stub) same native) not (generat$3 near3 (instrument$5 or hook) same native)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 14:26

S18	2	"6658416".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 16:31
S19	1048	(((native or executable or machine or binary or compiled) adj code) or binaries) and (access$3 or modify$3 or modification or alter$5 updat$3) near3 (field or value or variable) and (instrument$5 or hook$3) and (java or bytecode or byte-code or (byte adj code) or JIT$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 16:37
S20	1236	(((native or executable or machine or binary or compiled) adj code) or binaries) and (assign$4 or access$3 or modify$3 or modification or alter$5 updat$3) near3 (field or value or variable) and (instrument$5 or hook$3) and (java or bytecode or byte-code or (byte adj code) or JIT$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 16:39
S21	88	(((native or executable or machine or binary or compiled) adj code) or binaries) same ((assign$4 or access$3 or modify$3 or modification or alter$5 updat$3) near3 (field or value or variable)) and (instrument$5 or hook$3) and (java or bytecode or byte-code or (byte adj code) or JIT$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 16:51
S22	101	(((native or executable or machine or binary or compiled) adj code) or binaries) same ((assign$4 or access$3 or modify$3 or modification or alter$5 or updat$3) near3 (field or value or variable)) and (instrument$5 or hook$3 or patch$3) and (java or bytecode or byte-code or (byte adj code) or JIT$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/16 17:05

S23	138	(((native or executable or machine or binary or compiled) adj code) or binaries or executable) same ((assign$4 or access$3 or modify$3 or modification or alter$5 or updat$3) near3 (field or value or variable)) and (instrument$5 or hook$3 or patch$3) and (java or bytecode or byte-code or (byte adj code) or JIT$4) and (watch or break or watchpoint or breakpoint or watch-point or break-point or monitor$3 or evaluat$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 15:06
S24	10	"6016466".URPN.	USPAT	OR	OFF	2004/02/16 17:27
S25	8	("4713791" | "4937780" | "5371689" | "5590342" | "5732272" | "5758184" | "5828881" | "5872909").PN.	USPAT	OR	OFF	2004/02/16 17:37
S26	12	("5367685" | "5465258" | "5539907" | "5590331" | "5668999" | "5768592" | "5768593" | "5842017" | "5905895" | "5909577" | "5995754" | "6170083").PN.	USPAT	OR	OFF	2004/02/16 17:42
S27	7	"6289506".URPN.	USPAT	OR	OFF	2004/02/16 17:47
S28	16	("5768593" | "5937195" | "5970249" | "5999734" | "6011916" | "6014518" | "6044221" | "6072950" | "6073159" | "6077313" | "6078744" | "6085035" | "6113651" | "6189141" | "6223339" | "6253373").PN.	USPAT	OR	OFF	2004/02/17 07:38
S29	8	"6078744".URPN.	USPAT	OR	OFF	2004/02/17 07:43
S30	4	("5179702" | "5430850" | "5471593" | "5794005").PN.	USPAT	OR	OFF	2004/02/17 07:46
S31	42	717/130.ccls. and (push$3 or pop$4 or stor$3 or retriev$3) near2 stack	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 15:08
S32	14	717/130.ccls. and (push$3 or pop$4 or stor$3 or retriev$3) near2 stack same (live$6 or global or state)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 15:11

S33	17	717/130.ccls. and (push$3 or pop$4 or stor$3 or retriev$3) near2 stack same (live$6 or global or state or context)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 15:59
S34	1375	insert$3 near2 stub	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 15:59
S35	1	(insert$3 adj code) near2 stub	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 16:00
S36	1	(insert$3 adj code) near4 stub	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 16:01
S37	470	code near4 stub	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 16:01
S38	105	code near4 stub and (instrument$3 or hook$3 or probe or probin)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 16:01
S39	106	code near4 stub and (instrument$3 or hook$3 or probe or probing)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 16:16
S40	54192	insert$3 near3 (instrument$3 or hook$3 or probe or probing)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 16:17
S41	575	insert$3 near3 (location or placement) near3 (instrument$3 or hook$3 or probe or probing)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/17 16:17
S42	1	patch$3 adj (bytecode or byte-code or (byte adj code))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/18 07:25

S43	30	(instrument$3 or patch$3) near3 (bytecode or byte-code or (byte adj code))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/18 07:26
S44	3	(instrument$3 or patch$3) near3 (bytecode or byte-code or (byte adj code)) same (breakpoint or break-point or (break adj point) or watch or monitor or watchpoint or watch-point)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/18 11:13
S45	2	"20010047510"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/18 11:13
S46	1	"20010047510" and register	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/18 16:39
S47	10	("5901315" "6078744" "6298477" "6637024" "6662358").pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/18 16:40
S48	4	("6186677" "5976249").pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/23 08:10
S49	4	("6186677" "5987249").pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/02/23 08:10
S50	4094	re-compil$5 or recompil$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2004/07/28 16:35
S51	229	(re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again))) and (watch or watchpoint or watch-point)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/28 16:36
S52	217	(re-compil$5 or recompil$5) and (watch or watchpoint or watch-point)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/28 16:37

S53	17	(re-compil$5 or recompil$5) and (watch or watchpoint or watch-point) and dynamic and (flag$4 near3 field)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/28 16:44
S54	35	(re-compil$5 or recompil$5) and (watch or watchpoint or watch-point) and dynamic and ((flag$4 or bit) near3 (field or value))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/28 16:41
S55	35	(re-compil$5 or recompil$5 or re-build$3 or rebuild$3) and (watch or watchpoint or watch-point) and dynamic and (flag$4 near3 field)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/28 16:44
S56	171	(re-compil$5 or recompil$5) and (watch or watchpoint or watch-point) and dynamic	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/29 07:48
S57	23	"5787245".URPN.	USPAT	OR	OFF	2004/07/28 16:50
S58	14	"5835701".URPN.	USPAT	OR	OFF	2004/07/28 17:18
S59	22	("3815103" | "4104718" | "4394731" | "4533997" | "4802165" | "4811347" | "4815025" | "4903194" | "4937736" | "4953084" | "5025366" | "5029078" | "5075842" | "5115499" | "5132972" | "5175828" | "5175837" | "5193180" | "5230070" | "5257381" | "5404499" | "5438670").PN.	USPAT	OR	OFF	2004/07/28 17:34
S60	13	"4811347".URPN.	USPAT	OR	OFF	2004/07/28 17:41
S61	80	(re-compil$5 or recompil$5) and (watch or watchpoint or watch-point) and dynamic and ((bytecode or byte-code or (byte adj code)) or native)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/29 08:49
S62	2	"20020073063"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/29 08:55
S63	2	"5901315".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2004/07/29 08:55

S64	40	((byte adj code) or byte-code or bytecode) same (native or jit or jit$4) and (generat$3 or creat$3 or produc$3 or provid$3) near3 (instrument$5 or prolog$2 or epilog$2)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/14 12:13
S65	49	((byte adj code) or byte-code or bytecode) same (native or jit or jit$4 or ("machine code")) and (generat$3 or creat$3 or produc$3 or provid$3) near3 (instrument$5 or prolog$2 or epilog$2)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 12:28
S66	150	class same (method or function) same (flag or guard or (access$3 near3 (enabl$3 or disabl$3 or inhibit$3 or permission or restrict$3 or permit$4 or bitwise))) near5 (field or variable)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 14:09
S67	65	class near5 (method or function) same (flag or guard or (access$3 near3 (enabl$3 or disabl$3 or inhibit$3 or permission or restrict$3 or permit$4 or bitwise))) near5 (field or variable)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 12:34
S68	18	("5940616").URPN.	USPAT	OR	OFF	2005/04/25 13:29
S69	187	class same (method or function) same (flag or guard or (access$3 near3 (enabl$3 or disabl$3 or inhibit$3 or permission or restrict$3 or permit$4 or bitwise or control$4))) near5 (field or variable)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 13:46
S70	37	S69 not S66	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 13:46
S71	45	class same (method or function) same (flag or guard or ((access$3 or modified or modify$3 or modification or updat$3) near3 (control$4 or policy or policies or enabl$3 or disabl$3 or inhibit$3 or permission or restrict$3 or permit$4 or bitwise))) near5 (field or variable) same (event or handl$3 or instrument$5 or hook)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 14:28
S72	509	watchpoint or (watch adj point)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 14:28

S73	207	(watch or watchpoint or (watch adj point)) and (flag or guard or ((access$3 or modified or modify$3 or modification or updat$3) near3 (control$4 or policy or policies or enabl$3 or disabl$3 or inhibit$3 or permission or restrict$3 or permit$4 or bitwise))) near5 (field or variable) same (event or handl$3 or instrument$5 or hook)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 14:29
S74	11	(watch or watchpoint or (watch adj point)) same (flag or guard or ((access$3 or modified or modify$3 or modification or updat$3) near3 (control$4 or policy or policies or enabl$3 or disabl$3 or inhibit$3 or permission or restrict$3 or permit$4 or bitwise))) near5 (field or variable) same (event or handl$3 or instrument$5 or hook)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 14:52
S75	14	("5404496" | "5430862" | "5493723" | "5526485" | "5544311" | "5564041" | "5644703" | "5715440" | "5752013" | "5754839" | "5857094" | "5978937" | "6314530" | "6453410").PN. OR ("6754856").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/25 14:35
S76	1423	(watch or watchpoint or (watch adj point)) near5 (instrument$5 or hook or handler)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 15:28
S77	4	(watch or watchpoint or (watch adj point)) near5 (instrument$5 or hook or handler) same (field or variable) and ((guard$3 or inhibit$3 or disabl$3 or filter or restrict$3 or Boolean or condition$2) near3 execut$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/25 15:32
S78	21	("5335344" | "5465258" | "5539907" | "5790858" | "5832271" | "5920689" | "5960198" | "6079032").PN. OR ("6216237").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/25 16:28
S79	1	"6754856".pn.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/25 16:29
S80	5	("5715440" "6332212" "5940616" "5978937" "6820252").pn.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/25 16:30

S81	17	jvmdi	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/30 14:39
S82	13	jvmdi and (watch or event or hook)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/30 14:40
S83	13	jvmdi and (watch or event or hook or handler)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/30 14:40
S84	12	jvmdi and (watch or event or hook or handler) and (field or variable)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/30 14:40
S85	12	jvmdi and (watch or event or hook or handler) and (field or variable) and (access$3 or modify$3 or modification)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/04/30 14:41
S86	5	("20020073063" | "5548717" | "5706502" | "5901315" | "6061518").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/30 14:56
S87	4304	recompil$5 or re-compil$5 or ((repeat$3 or again or second or subsequent$2) near2 compil$5) and (java or jvm or bytecode or byte-code or (byte adj code))	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/30 14:58
S88	4049	recompil$5 or re-compil$5 or ((repeat$3 or again or second or subsequent$2) near2 compil$5) same (java or jvm or bytecode or byte-code or (byte adj code))	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/30 14:58
S89	4007	recompil$5 or re-compil$5 or ((repeat$3 or again or second or subsequent$2) near2 compil$5) same (java or jvm or bytecode or byte-code or (byte adj code)) and (instrument$5 or hook or handler)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/30 14:59
S90	3990	recompil$5 or re-compil$5 or ((repeat$3 or again or second or subsequent$2) near2 compil$5) same (java or jvm or bytecode or byte-code or (byte adj code)) same (instrument$5 or hook$3 or handl$3)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/04/30 14:59

S91	241	debug$4 and (re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again))) and (watch or watchpoint or value or variable) and native and (instrument$5 or hook$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/30 15:00
S92	418	debug$4 and (re-compil$5 or recompil$5 or (compil$5 near3 (repeat$7 or again))) and (watch or watchpoint or value or variable) and native	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/30 15:00
S93	392	S90 and S92	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/30 15:00
S94	34	S90 and S92 and (jvm or jvmdi or (debug adj interface))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/30 15:01

Terms used **java field access** and **guard**

Found **34,871** of 154

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Type inference for atomicity

Cormac Flanagan, Stephen N. Freund, Marina Lifshin

January 2005 **Proceedings of the 2005 ACM SIGPLAN international workshop on Types in languages design and implementation**

Full text available: pdf(168.02 KB) Additional Information: full citation, abstract, references, index terms

Atomicity is a fundamental correctness property in multithreaded programs. This paper presents an algorithm for verifying atomicity via type inference. The underlying type system supports guarded, write-guarded, and unguarded fields, as well as thread-local data, parameterized classes and methods, and protected locks. We describe an implementation of this algorithm for Java and discuss its performance and usability on benchmarks totaling sixty thousand lines of code.

Keywords: atomicity, concurrency, reduction, type inference

2 Recompilation for debugging support in a JIT-compiler

Mustafa M. Tikir, Jeffrey K. Hollingsworth, Guei-Yuan Lueh

November 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**, Volume 28 Issue 1

Full text available: pdf(89.55 KB) Additional Information: full citation, abstract, references, index terms

A static Java compiler converts Java source code into a verifiably secure and compact architecture-neutral intermediate format, called Java *byte codes*. The Java byte codes can be either interpreted by a Java Virtual Machine or translated into native code by Java Just-In-Time compilers. Static Java compilers embed debug information in the Java class files to be used by the source level debuggers. However, the debug information is generated for architecture independent byte codes and most o ...

Keywords: Java, Java virtual machine debugger interface, debug information, dynamic recompilation, field access watch, just-in-time compilation

3 Type-safe multithreading in cyclone
Dan Grossman
January 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN international workshop on Types in languages design and implementation,** Volume 38 Issue 3
Full text available: pdf(228.33 KB) Additional Information: full citation, abstract, references, citings, index terms

We extend Cyclone, a type-safe polymorphic language at the C level of abstraction, with threads and locks. Data races can violate type safety in Cyclone. An extended type system statically guarantees their absence by enforcing that thread-shared data is protected via locking and that thread-local data does not escape the thread that creates it. The extensions interact smoothly with parametric polymorphism and region-based memory management. We present a formal abstract machine that models the ne ...

Keywords: cyclone, data races, types

4 Core semantics of multithreaded Java
Jeremy Manson, William Pugh
June 2001 **Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande**
Full text available: pdf(816.27 KB) Additional Information: full citation, abstract, references, citings, index terms

Java has integrated multithreading to a far greater extent than most programming languages. It is also one of the only languages that specifies and requires safety guarantees for improperly synchronized programs. It turns out that understanding these issues is far more subtle and difficult than was previously thought. The existing specification makes guarantees that prohibit standard and proposed compiler optimizations; it also omits guarantees that are necessary for safe execution of much ex ...

5 Onward!: Finding bugs is easy
David Hovemeyer, William Pugh
October 2004 **Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications**
Full text available: pdf(104.93 KB) Additional Information: full citation, abstract, references, index terms

Many techniques have been developed over the years to automatically find bugs in software. Often, these techniques rely on formal methods and sophisticated program analysis. While these techniques are valuable, they can be difficult to apply, and they aren't always effective in finding real bugs.

<i>Bug patterns</i> are code idioms that are often errors. We have implemented automatic detectors for a variety of bug patterns found in Java programs. In this extended abstract<s ...

Keywords: bug checkers, bug patterns, bugs, static analysis

6 Efficient and precise datarace detection for multithreaded object-oriented programs
Jong-Deok Choi, Keunwoo Lee, Alexey Loginov, Robert O'Callahan, Vivek Sarkar, Manu Sridharan

May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(171.13 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a novel approach to dynamic datarace detection for multithreaded object-oriented programs. Past techniques for on-the-fly datarace detection either sacrificed precision for performance, leading to many false positive datarace reports, or maintained precision but incurred significant overheads in the range of 3x to 30x. In contrast, our approach results in very few false positives and runtime overhead in the 13% to 42% range, making it both efficient *and* precis ...

Keywords: dataraces, debugging, multithreaded programming, object-oriented programming, parallel programs, race conditions, static-dynamic co-analysis, synchronization

7 Constructing compact models of concurrent Java programs
James C. Corbett
March 1998 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis**, Volume 23 Issue 2

Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, references, citings, index terms

Finite-state verification technology (e.g., model checking) provides a powerful means to detect concurrency errors, which are often subtle and difficult to reproduce. Nevertheless, widespread use of this technology by developers is unlikely until tools provide automated support for extracting the required finite-state models directly from program source. In this paper, we explore the extraction of compact concurrency models from Java code. In particular, we show how static pointer analysis, whic ...

Keywords: finite-state verification, model extraction, static analysis

8 A type and effect system for atomicity
Cormac Flanagan, Shaz Qadeer
May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation**, Volume 38 Issue 5

Full text available: pdf(266.52 KB) Additional Information: full citation, abstract, references, citings, index terms

Ensuring the correctness of multithreaded programs is difficult, due to the potential for unexpected and nondeterministic interactions between threads. Previous work addressed this problem by devising tools for detecting *race conditions*, a situation where two threads simultaneously access the same data variable, and at least one of the accesses is a write. However, verifying the absence of such simultaneous-access race conditions is neither necessary nor sufficient to ensure the absence o ...

Keywords: atomicity, multithreading, race conditions, static checking

9
Technical papers: concurrency: Assuring and evolving concurrent programs: annotations and policy

Aaron Greenhouse, William L. Scherlis
May 2002 **Proceedings of the 24th International Conference on Software Engineering**
Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

Assuring and evolving concurrent programs requires understanding the concurrency-related design decisions used in their implementation. In Java-style shared-memory programs, these decisions include which state is shared, how access to it is regulated, the roles of threads, and the policy that distinguishes desired concurrency from race conditions. These decisions rarely have purely local manifestations in code.In this paper, we use case studies from production Java code tc explore the costs and ...

10 Ada, C, C++, and Java vs. the Steelman
David A. Wheeler
July 1997 **ACM SIGAda Ada Letters**, Volume XVII Issue 4
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, citings, index terms

This paper compares four computer programming languages (Ada95, C, C++, and Java) with the requirements of "Steelman", the original 1978 requirements document for the Ada computer programming language. This paper provides a view of the capabilities of each of these languages, and should help those trying to understand their technical similarities, differences, and capabilities.

11 Zones, contracts and absorbing changes: an approach to software evolution
Huw Evans, Peter Dickman
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(2.46 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes a novel approach to managing the evolution of distributed, persistent systems at run-time. This is achieved by partitioning a system into disjoint zones, each of which can be evolved without affecting code in any other. Contracts are defined between zones, making type-level interdependencies and inter-zone communication explicit. Programmer supplied code is added to the running system, at the boundary between zones, to constrain the sco ...

12 Using shape analysis to reduce finite-state models of concurrent Java programs
James C. Corbett
January 2000 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 9 Issue 1
Full text available: pdf(284.92 KB) Additional Information: full citation, abstract, references, citings, index terms

Finite-state verification (e.g., model checking) provides a powerful means to detect concurrency errors, which are often subtle and difficult to reproduce. Nevertheless, widespread use of this technology by developers is unlikely until tools provide automated support for extracting the required finite-state models directly from program source. Unfortunately, the dynamic features of modern languages such as Java complicate the construction of compact finite-state models for verification. I ...

Keywords: Java, concurrent systems, finite-state verification, model extraction, modeling, shape analysis, state-space reductions

13 Cloning-based context-sensitive pointer alias analysis using binary decision diagrams
John Whaley, Monica S. Lam
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation,** Volume 39 Issue 6
Full text available: pdf(277.87 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the first scalable context-sensitive, inclusion-based pointer alias analysis for Java programs. Our approach to context sensitivity is to create a clone of a method for every context of interest, and run a *context-insensitive* algorithm over the expanded call graph to get *context-sensitive* results. For precision, we generate a clone for every acyclic path through a program's call graph, treating methods in a strongly connected component as a single node. Normally ...

Keywords: Datalog, Java, binary decision diagrams, cloning, context-sensitive, inclusion-based, logic programming, pointer analysis, program analysis, scalable

14 Improving the Java memory model using CRF
Jan-Willem Maessen, Xiaowei Shen
October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications,** Volume 35 Issue 10
Full text available: pdf(193.84 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes alternative memory semantics for Java programs using an enriched version ol the Commit/Reconcile/Fence (CRF) memory model [16]. It outlines a set of reasonable practices for safe multithreaded programming in Java. Our semantics allow a number of optimizations such as load reordering that are currently prohibited. Simple thread-local algebraic rules express the effects of optimizations at the source or bytecode level. The rules focus on reordering source-level operations; the ...

Keywords: Java, commit/reconcile/fence, compilation, memory models

15 Implementing jalapeño in Java
Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Barton, Susan Flynn Hummel, Janice C. Sheperd, Mark Mergen
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications,** Volume 34 Issue 10
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

Jalapeño is a virtual machine for Java™ servers written in Java.A running Java program involves four layers of functionality: the user code, the virtual-machine, the operating system, and the

hardware. By drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño reduces the boundary-crossing overhead and opens up more opportunities for optimization.To get Jalapeño started, a boot image of a ...

16 Practical predicate dispatch
Todd Millstein
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10
Full text available: pdf(192.02 KB) Additional Information: full citation, abstract, references, citings, index terms

<i>Predicate dispatch</i> is an object-oriented (OO) language mechanism for determining the method implementation to be invoked upon a message send. With predicate dispatch, each method implementation includes a predicate guard specifying the conditions under which the method should be invoked, and logical implication of predicates determines the method overriding relation. Predicate dispatch naturally unifies and generalizes several common forms of dynamic dispatch, including traditi ...

Keywords: dynamic dispatch, modular typechecking, predicate dispatch

17 Technical papers: program analysis: Specifying multithreaded Java semantics for program verification
Abhik Roychoudhury, Tulika Mitra
May 2002 **Proceedings of the 24th International Conference on Software Engineering**
Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references, citings, index terms

The Java programming language supports multithreading where the threads interact among themselves via read/write of shared data. Most current work on multithreaded Java program verification assumes a model of execution that is based on interleaving of the operations of the individual threads. However, the Java language specification (which any implementations of Java multithreading must follow) supports a weaker model of execution, called the Java Memory Model (JMM). The JMM allows certain reord ...

18 New techniques for security and reliability enhancement in embedded systems: Analyzing heap error behavior in embedded JVM environments
G. Chen, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, M. J. Irwin
September 2004 **Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis**
Full text available: pdf(285.76 KB) Additional Information: full citation, abstract, references, index terms

Recent studies have shown that transient hardware errors caused by external factors such as alpha particles and cosmic ray strikes can be responsible for a large percentage of system down-time. Denser processing technologies, increasing clock speeds, and low supply voltages used in embedded systems can worsen this problem. In many embedded environments, one may not want to provision extensive error protection in hardware because of (i) form-factor or power consumption limitations, and/or (ii) to ...

Keywords: JVM, softerrors

19 Type-based race detection for Java
Cormac Flanagan, Stephen N. Freund
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(237.37 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a static race detection analysis for multithreaded Java programs. Our analysis is based on a formal type system that is capable of capturing many common synchronization patterns. These patterns include classes with internal synchronization, classes thatrequire client-side synchronization, and thread-local classes. Experience checking over 40,000 lines of Java code with the type system demonstrates that it is an effective approach for eliminating races conditions. On lar ...

20 Stack allocation and synchronization optimizations for Java using escape analysis
Jong-Deok Choi, Manish Gupta, Mauricio J. Serrano, Vugranam C. Sreedhar, Samuel P. Midkiff
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Full text available: pdf(632.85 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an *escape analysis* framework for Java to determine (1) if an object is not reachable after its method of creation returns, allowing the object to be allocated on the stack, and (2) if an object is reachable only from a single thread during its lifetime, allowing unnecessary synchronization operations on that object to be removed. We introduce a new program abstraction for escape analysis, the *connection graph*, that is used to establish reachability relationshi ...

Keywords: Connection graphs, escape analysis, points-to graph

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Terms used **java field access** and **instrument**

Found **37,399** of 154

Sort results by relevance

Display results expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200

Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Hybrid dynamic data race detection
Robert O'Callahan, Jong-Deok Choi
June 2003 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 38 Issue 10
Full text available: pdf(158.47 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a new method for dynamically detecting potential data races in multithreaded programs. Our method improves on the state of the art in accuracy, in usability, and in overhead. We improve accuracy by combining two previously known race detection techniques -- *lockset-based detection and happens-before-based detection* -- to obtain fewer false positives than lockset-based detection alone. We enhance usability by reporting more information about detected races than any previous dyna ...

Keywords: Java, dynamic race detection, happens-before, lockset hybrid

2 Recompilation for debugging support in a JIT-compiler
Mustafa M. Tikir, Jeffrey K. Hollingsworth, Guei-Yuan Lueh
November 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**, Volume 28 Issue 1
Full text available: pdf(89.55 KB) Additional Information: full citation, abstract, references, index terms

A static Java compiler converts Java source code into a verifiably secure and compact architecture-neutral intermediate format, called Java *byte codes*. The Java byte codes can be either interpreted by a Java Virtual Machine or translated into native code by Java Just-In-Time compilers. Static Java compilers embed debug information in the Java class files to be used by the source level debuggers. However, the debug information is generated for architecture independent byte codes and most o ...

Keywords: Java, Java virtual machine debugger interface, debug information, dynamic recompilation, field access watch, just-in-time compilation

3 Cache-conscious structure definition
Trishul M. Chilimbi, Bob Davidson, James R. Larus
May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation**, Volume 34 Issue 5
Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, citings, index terms

A program's cache performance can be improved by changing the organization and layout of its data---even complex, pointer-based data structures. Previous techniques improved the cache performance of these structures by arranging distinct instances to increase reference locality. These techniques produced significant performance improvements, but worked best for small structures that could be packed into a cache block.This paper extends that work by concentrating on the internal organization of f ...

Keywords: cache-conscious definition, class splitting, field reorganization, structure splitting

4 Characterizing the memory behavior of Java workloads: a structured view and opportunities for optimizations
Yefim Shuf, Mauricio J. Serrano, Manish Gupta, Jaswinder Pal Singh
June 2001 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 29 Issue 1
Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings

This paper studies the memory behavior of important Java workloads used in benchmarking Java Virtual Machines (JVMs), based on instrumentation of both application and library code in a state-of-the-art JVM, and provides structured information about these workloads to help guide systems' design. We begin by characterizing the inherent memory behavior of the benchmarks, such as information on the breakup of heap accesses among different categories and on the hotness of references to fields and met ...

5 Diverse topics: Field level analysis for heap space optimization in embedded java environments
Guangyu Chen, Mahmut Kandemir, N. Vijaykrishnan, Mary Janie Irwin
October 2004 **Proceedings of the 4th international symposium on Memory management**
Full text available: pdf(1.67 MB) Additional Information: full citation, abstract, references, index terms

Memory constraint presents one of the critical challenges for embedded software writers. While circuit-level solutions based on cramming as many bits as possible into the smallest area possible are certainly important, memory-conscious software can bring much higher benefits. Focusing on an embedded Java-based environment, this paper studies potential benefits and challenges when heap memory is managed at a field granularity instead of object. This paper discusses these benefits and challenge ...

Keywords: garbage collection, java virtual machine

6 Error-free garbage collection traces: how to cheat and not get caught
Matthew Hertz, Stephen M Blackburn, J Eliot B Moss, Kathryn S. McKinley, Darko Stefanović
June 2002 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 30 Issue 1
Full text available: pdf(105.06 KB) Additional Information: full citation, abstract, references, citings

Programmers are writing a large and rapidly growing number of programs in object-oriented languages such as Java that require garbage collection (GC). To explore the design and evaluation of GC algorithms quickly, researchers are using simulation based on traces of object allocation and lifetime behavior. The *brute force* method generates perfect traces using a whole-heap GC at every potential GC point in the program. Because this process is prohibitively expensive, researchers often use < ...

7 Technical papers: dynamic program analysis: Tracking down software bugs using automatic anomaly detection
Sudheendra Hangal, Monica S. Lam
May 2002 **Proceedings of the 24th International Conference on Software Engineering**
Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper introduces DIDUCE, a practical and effective tool that aids programmers in detecting complex program errors and identifying their root causes. By instrumenting a program and observing its behavior as it runs, DIDUCE dynamically formulates hypotheses of invariants obeyed by the program. DIDUCE hypothesizes the strictest invariants at the beginning, and gradually relaxes the hypothesis as violations are detected to allow for new behavior. The violations reported help users to catch soft ...

8 Atomizer: a dynamic atomicity checker for multithreaded programs
Cormac Flanagan, Stephen N Freund
January 2004 **ACM SIGPLAN Notices , Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages**, Volume 39 Issue 1
Full text available: pdf(195.13 KB) Additional Information: full citation, abstract, references, citings, index terms

Ensuring the correctness of multithreaded programs is difficult, due to the potential for unexpected interactions between concurrent threads. Much previous work has focused on detecting race conditions, but the absence of race conditions does not by itself prevent undesired thread interactions. We focus on the more fundamental non-interference property of *atomicity*; a method is atomic if its execution is not affected by and does not interfere with concurrently-executing threads. Atomic me ...

Keywords: atomicity, dynamic analysis, reduction

9 Practical experience with an application extractor for Java
Frank Tip, Chris Laffra, Peter F. Sweeney, David Streeter
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(2.31 MB) Additional Information: full citation, abstract, references, citings, index terms

Java programs are routinely transmitted over low-bandwidth network connections as compressed class file archives (i.e., zip files and jar files). Since archive size is directly proportional to download time, it is desirable for applications to be as small as possible. This paper is concerned with the use of program transformations such as removal of dead methods and fields, inlining of method calls, and simplification of the class hierarchy for reducing application size. Such "extract ...

10 Korat: automated testing based on Java predicates
Chandrasekhar Boyapati, Sarfraz Khurshid, Darko Marinov
July 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis**, Volume 27 Issue 4
Full text available: pdf(171.43 KB) Additional Information: full citation, abstract, references, citings

This paper presents Korat, a novel framework for automated testing of Java programs. Given a formal specification for a method, Korat uses the method precondition to automatically generate all (nonisomorphic) test cases up to a given small size. Korat then executes the method on each test case, and uses the method postcondition as a test oracle to check the correctness of each output.To generate test cases for a method, Korat constructs a Java predicate (i.e., a method that returns a boolean) fr ...

11 A framework for reducing the cost of instrumented code
Matthew Arnold, Barbara G. Ryder
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5
Full text available: pdf(1.78 MB) Additional Information: full citation, abstract, references, citings, index terms

Instrumenting code to collect profiling information can cause substantial execution overhead. This overhead makes instrumentation difficult to perform at runtime, often preventing many known *offline* feedback-directed optimizations from being used in online systems. This paper presents a general framework for performing *instrumentation sampling* to reduce the overhead of previously expensive instrumentation. The framework is simple and effective, using code-duplication and *coun ...*

12 Middleware performance analysis: Performance monitoring of java applications
M. Harkema, D. Quartel, B. M. M. Gijsen, R. D. van der Mei
July 2002 **Proceedings of the third international workshop on Software and performance**
Full text available: pdf(219.69 KB) Additional Information: full citation, abstract, references, citings, index terms

Over the past few years, Java has evolved into a mature platform for developing enterprise applications. A critical factor for the commercial success of these applications is end-to-end performance, e.g., in terms of response times, throughput and availability. This raises the need for the development, validation and analysis of performance models to predict performance metrics of interest. To develop and validate performance models, insight in the execution behavior of the application is essent ...

Keywords: performance measurement and monitoring of java applications

13 Efficient and precise datarace detection for multithreaded object-oriented programs
Jong-Deok Choi, Keunwoo Lee, Alexey Loginov, Robert O'Callahan, Vivek Sarkar, Manu Sridharan
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5
Full text available: pdf(171.13 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a novel approach to dynamic datarace detection for multithreaded object-oriented programs. Past techniques for on-the-fly datarace detection either sacrificed precision for performance, leading to many false positive datarace reports, or maintained precision but incurred significant overheads in the range of 3x to 30x. In contrast, our approach results in very few false positives and runtime overhead in the 13% to 42% range, making it both efficient *and* precis ...

Keywords: dataraces, debugging, multithreaded programming, object-oriented programming, parallel programs, race conditions, static-dynamic co-analysis, synchronization

14 Object equality profiling
Darko Marinov, Robert O'Callahan
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(577.47 KB) Additional Information: full citation, abstract, references, citings, index terms

We present *Object Equality Profiling* (OEP), a new technique for helping programmers discover optimization opportunities in programs. OEP discovers opportunities for replacing a set of equivalent object instances with a single representative object. Such a set represents an opportunity for automatically or manually applying optimizations such as hash consing, heap compression, lazy allocation, object caching, invariant hoisting, and more. To evaluate OEP, we implemented a tool to help prog ...

Keywords: Java language, object equality, object mergeability, profile-guided optimization, profiling, space savings

15 Poster session: RAIL: code instrumentation for .NET
Bruno Cabral, Paulo Marques, Luís Silva
October 2004 **Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications**
Full text available: pdf(150.85 KB) Additional Information: full citation, abstract, references, index terms

Code instrumentation is a mechanism that allows modules of programs to be completely rewritten at runtime. With the advent of virtual machines, this type of functionality is becoming more and more interesting because it allows the introduction of new functionality after an application has been deployed, easy implementation of aspect-oriented programming, performing security verifications, dynamic software upgrading, among others.

The Runtime Assembly Instrumentation Library (RAIL) is o ...

Keywords: .NET platform, code instrumentation

16 Static conflict analysis for multi-threaded object-oriented programs
Christoph von Praun, Thomas R. Gross
May 2003 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation**, Volume 38 Issue 5
Full text available: pdf(674.11 KB) Additional Information: full citation, abstract, references, citings, index terms

A compiler for multi-threaded object-oriented programs needs information about the sharing of objects for a variety of reasons: to implement optimizations, to issue warnings, to add instrumentation to detect access violations that occur at runtime. An Object Use Graph (OUG) statically captures accesses from different threads to objects. An OUG extends the Heap Shape Graph (HSG), which is a compile-time abstraction for runtime objects (nodes) and their reference relations (edges). An OUG specifie ...

Keywords: heap shape graph, object use graph, program analysis, race detection, representations for concurrent programs

17 Testing of java web services for robustness
Chen Fu, Barbara G. Ryder, Ana Milanova, David Wonnacott
July 2004 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis**, Volume 29 Issue 4
Full text available: pdf(264.32 KB) Additional Information: full citation, abstract, references, index terms

This paper presents a new compile-time analysis that enables a testing methodology for white-box coverage testing of error recovery code (i.e., exception handlers) in Java web services using compiler-directed fault injection. The analysis allows compiler-generated instrumentation to guide the fault injection and to record the recovery code exercised. (An injected fault is experienced as a Java exception.) The analysis (i) identifies the *exception-flow 'def-uses'* to be tested in this manne ...

Keywords: def-use testing, exceptions, java, test coverage metrics

18 Vertical profiling: understanding the behavior of object-priented applications
Matthias Hauswirth, Peter F. Sweeney, Amer Diwan, Michael Hind
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10
Full text available: pdf(1.16 MB) Additional Information: full citation, abstract, references, index terms

Object-oriented programming languages provide a rich set of features that provide significant software engineering benefits. The increased productivity provided by these features comes at a

justifiable cost in a more sophisticated runtime system whose responsibility is to implement these features efficiently. However, the virtualization introduced by this sophistication provides a significant challenge to understanding complete system performance, not found in traditionally compiled languages ...

Keywords: hardware performance monitors, perturbation, software performance monitors, vertical profiling, whole-system analysis

19 Related field analysis
Aneesh Aggarwal, Keith H. Randall
May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5
Full text available: pdf(843.74 KB) Additional Information: full citation, abstract, references, citings, index terms

We present an extension of field analysis (sec [4]) called *related field analysis* which is a general technique for proving relationships between two or more fields of an object. We demonstrate the feasibility and applicability of related field analysis by applying it to the problem of removing array bounds checks. For array bounds check removal, we define a pair of related fields to be an integer field and an array field for which the integer field has a known relationship to the lengt ...

20 Practical extraction techniques for Java
Frank Tip, Peter F. Sweeney, Chris Laffra, Aldo Eisma, David Streeter
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Full text available: pdf(1.01 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Reducing application size is important for software that is distributed via the internet, in order to keep download times manageable, and in the domain of embedded systems, where applications are often stored in (Read-Only or Flash) memory. This paper explores extraction techniques such as the removal of unreachable methods and redundant fields, inlining of method calls, and transformation of the class hierarchy for reducing application size. We implemented a number of extraction techniques in < ...

Keywords: Application extraction, call graph construction, class hierarchy transformation, packaging, whole-program analysis

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Web [Images](#) [Groups](#) [News](#) [Froogle](#) [Local](#) **more »**

java field access + instrument* Search [Advanced Search](#) [Preferences](#)

Web Results **1** - **10** of about **537,000** for **java field access** + **instrument***. (**0.34** seconds)

Java programming dynamics, Part 6: Aspect-oriented changes with ...
... how the Javassist search-and-replace support makes editing **Java** bytecode practically ... **field access** transformations, and a new object transformation. ...
www-106.ibm.com/developerworks/ **java**/library/j-dyn0302.html - 61k - May 21, 2005 - Cached - Similar pages

[PPT] Recompilation for Debugging Support in a JIT-Compiler
File Format: Microsoft Powerpoint 97 - View as HTML
... Generate debug information by dynamically re-compiling the related **Java** ...
Field access and modification points. Offset to the **instrumentation** code for ...
www.cs.umd.edu/~tikir/presentations/paste02.ppt - Similar pages

Javassist: **Java** Bytecode Engineering Made Simple @ JDJ
... allow you to do things like replace **field access**, redirect method calls, ...
Javassist is a powerful **Java** library that helps provide **instrumentation** of ...
java.sys-con.com/read/38672.htm - 66k - Cached - Similar pages

Open Source Project for a
Network Data **Access** Protocol -- 7 ...
... It has OPeNDAP data **access** to datasets compatible with the **Java**-NetCDF API,
... **Field** chosen for indexing/sorting is not always **instrument**/profile ID ...
www.po.gso.uri.edu/tracking/ meetings/tech03/report-html/report_8.html - 12k - Cached - Similar pages

[#CG-576] Member **access** violation from **instrumentation** classes ...
... Member **access** violation from **instrumentation** classes ... As a workaround either use a non-tracing breakpoint or use **java**.lang.reflect.**Field**. ...
jira.omnicore.com/secure/ViewIssue.jspa?key=CG-576 - 18k - Cached - Similar pages

Java Wallet Set Up
... download the **Java** Wallet in the **Java** Developer Connection's Early **Access** section.
... you must configure at least one payment **instrument** in your Wallet. ...
java.sun.com/products/commerce/user_guide.html - 18k - Cached - Similar pages

Powers of 10 - Interactive **Java** Tutorial
... buildings of the National High Magnetic **Field** Laboratory in Tallahassee, Florida.
... Please install this software in order to view our interactive **Java** ...
micro.magnet.fsu.edu/primer/ **java**/scienceopticsu/powersof10/ - 34k - May 21, 2005 - Cached - Similar pages

Java based, complete interface engine offering improved data ...
... **Java** software based device that contains the **instrument** communication ...
Enhanced Microbiology Interface captures every data **field** an **instrument** is ...
www.dawning.com/products/jresultnet/micro_module.html - 32k - Cached - Similar pages

Interactive Programming In **Java**
... Libraries -- such as the **Java** source code or the cs101 distribution -- generally

... looks like package syntax (or **field access** syntax), but it is not. ...
www.cs101.org/ipij/procedures.html - 61k - Cached - Similar pages

Annotated Declare Error/Warning
... To declare a warning, you annotate a **field** with @DeclareWarning. ... DAO classes should not **access** the Driver class [**java**] Car DAO save [**java**] WARNING: ...
docs.jboss.org/aop/1.1/aspect-framework/ examples/annotated-declare/annotated-declare.html - 14k - Cached - Similar pages

Goooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

Free! Google Desktop Search: Search your own computer. Download now.

Find: emails - files - chats - web history - media - PDF

java field access + instrument* Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

Welcome United States Patent and Trademark Office

Search Results

Results for "((java<in>metadata) <and> (field access* <in>metadata) <and> (guard*&l..." e-mail
Your search matched **0** of **1160635** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» View Session History

» New Search

Modify Search

((java<in>metadata) <and> (field access* <in>metadata) <and> (guard*<in>r

☐ Check to search only within this results set

Display Format: ◉ Citation ○ Citation & Abstract

» **Key**

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you assistance revising your search.